МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия
Вариант 13

Студент гр. 8304 _____ Масалыкин Д.Р.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++ в фреймворке Qt.

Постановка задачи.

1) проанализировать полученное задание, выделив рекурсивно определяемые информационные объекты и (или) действия;
2) разработать программу, использующую рекурсию;
3) сопоставить рекурсивное решение с итеративным решением задачи;
4) сделать вывод о целесообразности и эффективности рекурсивного решения данной задачи.

Задание.

Вариант 13: Построить синтаксический анализатор для понятия скобки.

скобки::=A|скобка скобки

скобка::= (B скобки)

Описание алгоритма.

Для вычисления значения выражения методом рекурсии необходимо считать строку, затем рекурсивно проверить корректность выражения.
При проверке на соответствие определению «скобки», проверяется либо наличие буквы «A», либо наличие конструкции «скобка скобки», где скобка это «(B скобки)». Алгоритм возвращает false если строка не соответствует определению, и true в ином случае. После завершение алгоритма итератор должен находиться в конце строки, в противном случае строка неккореткна.

```
проверить_выражение(строка, позиция, счетчик_скобок, флаг)
        если текущий_элемент «A» и позиция == длина -1
                вернуть true
если не флаг
                если текущий_элемент = ”(”
```

```
                позиция += 1
                счетчик_скобок += 1
                если текущий_элемент == «B»
                    позиция += 1
                    вернуть проверить_выражение(строка, позиция, счетчик_скобок, флаг)
                иначе
                    вернуть false
            иначе
                вернуть false
            если текущий_элемент == «A»
                флаг = true
                позиция += 1
                вернуть проверить_выражение(строка, позиция, счетчик_скобок, флаг)
        иначе
            если текущий_элемент == «)»
                позиция += 1
            иначе
                вернуть false
            если текущий_элемент != «A»
                флаг = false
                вернуть проверить_выражение(строка, позиция, счетчик_скобок, флаг)
            вернуть false
```

Спецификация программы.

Программа предназначена для синтаксического анализа выражения методом рекурсии.

Программа написана на языке C++. Входными данными является строка. Выходными данными являются конечный результат (корректна строка или нет). Данные выводятся в QMessageBox.

Тестирование.

Input	Output
"A"	true
"(B(B(B(BA))))A"	true
""	false
"AAA"	false
"()"	false
"(B(B(B(B(BA))))))(B(B(BA)))A"	true

Анализ алгоритма.

Алгоритм работает за линейное время. Недостаток рекурсивного алгоритма – ограниченный стек вызовов функций, что в свою очередь накладывает ограничение на размер входной строки, а также затраты производительности на вызов функций.

Описание функций и СД

Функции, реализующие интерфейс синтаксического анализатора.

```
bool is_brackets(std::string inp, unsigned long long i, int br_cntr, bool flag)
```

Метод класса, который принимает на вход строку, текущее положение и флаг(для обозначения конца открывающих круглых скобок). Возвращаемое значение: true если выражение корректно, false – если нет. Реализует алгоритм синтаксической проверки выражения.

Выводы.

В ходе работы были получены навыки реализации рекурсивных алгоритмов и проанализирована их целесообразность для задачи анализа выражения. Рекурсия является не эффективным методом анализа выражения.

Приложение А.

mainwondow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QMessageBox>
#include <iostream>
#include <string>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:

    void on_pushButton_2_clicked();

    void on_pushButton_clicked();

    void on_pushButton_3_clicked();

public:
    bool is_brackets(std::string inp, unsigned long long i, int br_cntr, bool flag);

private:
    Ui::MainWindow *ui;
    std::string res;
};

#endif // MAINWINDOW_H
```

main.cpp

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setWindowTitle("LR_1 var 13");
    w.show();

    return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_2_clicked()
{
    QApplication::quit();
}

bool MainWindow::is_brackets(std::string inp, unsigned long long i, int br_cntr, bool flag){
    if(i == inp.length() - 1){
        if(inp[i] == 'A'){
            return true;
        }
        else{
            return false;
        }
    }
    if(inp[i] == 'A' && i != inp.length() -1 && flag)
        return false;
    if(!flag){
        if(inp[i] == '('){
            i++;
            br_cntr++;
            if(inp[i]  == 'B'){
                i++;
                return is_brackets(inp, i, br_cntr, flag);
            }
            else{
            return false;
            }
        }
        else if(inp[i] == 'A'){
            i++;
            flag = true;
            return is_brackets(inp, i, br_cntr, flag);
        }
            else{
            return false;
        }

    }
    else{
        if(inp[i] == ')'){
            if(br_cntr > 0){
                i++;
                return is_brackets(inp, i, br_cntr -1, flag);
            }
            else{
                return false;
            }
        }
        if(inp[i] != 'A'){
            flag = false;
            return is_brackets(inp, i, br_cntr, flag);
        }
    }
    return false;
}

void MainWindow::on_pushButton_clicked()
{
    if(is_brackets("A", 0, 0 ,false) == true){
         if (is_brackets("(B(B(B(BA))))A", 0, 0 ,false) == true) {
             if (is_brackets("", 0, 0 ,false) == false){
                 if(is_brackets("AAA", 0, 0 ,false) == false){
                     if(is_brackets("()", 0, 0 ,false) == false){
                        if(is_brackets("(B(B(B(B(BA))))))(B(B(BA)))A", 0, 0 ,false) == true)
```

```
                        QMessageBox::information(this, "Result", "Tests passed");
                    }
                }
            }
        }
    }
}

void MainWindow::on_pushButton_3_clicked()
{
    res = ui->lineEdit->text().toStdString();
    if(res == "AA"){
        QMessageBox::warning(this, "Result", "NOT EXPRESSION");
        return;
    }
    if(is_brackets(res, 0, 0, false)){
        QMessageBox::information(this, "Result", "EXPRESSION");
    }
    else{
        QMessageBox::warning(this, "Result", "NOT EXPRESSION");
    }
}
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QLineEdit" name="lineEdit">
    <property name="geometry">
     <rect>
      <x>80</x>
      <y>50</y>
      <width>301</width>
      <height>22</height>
     </rect>
    </property>
   </widget>
   <widget class="QLabel" name="label">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>60</y>
      <width>55</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>Enter text!</string>
    </property>
   </widget>
   <widget class="QPushButton" name="pushButton">
    <property name="geometry">
     <rect>
      <x>240</x>
      <y>140</y>
      <width>93</width>
      <height>28</height>
     </rect>
    </property>
    <property name="text">
     <string>Test</string>
```

```
    </property>
   </widget>
   <widget class="QPushButton" name="pushButton_2">
    <property name="geometry">
     <rect>
      <x>60</x>
      <y>140</y>
      <width>91</width>
      <height>28</height>
     </rect>
    </property>
    <property name="text">
     <string>Close</string>
    </property>
   </widget>
   <widget class="QPushButton" name="pushButton_3">
    <property name="geometry">
     <rect>
      <x>140</x>
      <y>190</y>
      <width>93</width>
      <height>28</height>
     </rect>
    </property>
    <property name="text">
     <string>Check</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>400</width>
     <height>26</height>
    </rect>
   </property>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```